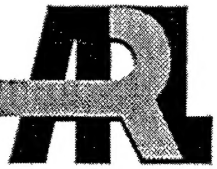
ARMY RESEARCH LABORATORY



# A Comparison of the Performance of Two Popular Symmetric Multiprocessors When Used to Run High Performance Computing Applications

by Daniel M. Pressel, Stephen Schraml, Steven Thompson, Dixie Hisley, Punyam Satya-narayana, Michael Knowles, and Darren M. Wah

ARL-TR-2476 March 2002

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

**Army Research Laboratory**
Aberdeen Proving Ground, MD 21005-5067

**ARL-TR-2476** **March 2002**

# A Comparison of the Performance of Two Popular Symmetric Multiprocessors When Used to Run High Performance Computing Applications

Daniel M. Pressel and Dixie Hisley
Computational and Information Sciences Directorate, ARL

Stephen Schraml
Weapons and Materials Research Directorate, ARL

Steven Thompson, Punyam Satya-narayana, and Michael Knowles
Raytheon Systems Company

Darren M. Wah
Major Shared Resource Center, ARL

# Abstract

Traditionally, symmetric multiprocessors have used modest numbers of processors. Since many of them were bus-based systems, they inherently lacked scalability to what might be referred to as moderate-sized systems. With the advent of the Sun HPC 10000 and the SGI Origin, we now have symmetric multiprocessors that have successfully scaled to moderate-sized systems. In fact, SGI has had some success at scaling the Origin into the lower end of the range of large systems. The first symmetric multiprocessor to make that claim was the Convex Exemplar. But based on our experience at the Distributed Center located at NRAD, San Diego, CA (now the Naval Command Control and Ocean Surveillance Center), its overall performance and scalability left something to be desired.

This report presents the results from runs involving a variety of programs on the SGI Origins and Sun HPC 10000s located at the U.S. Army Research Laboratory (ARL)-**MSRC**, the Naval Research Laboratory (NRL-**DC**), Washington, DC, and other places. Some of these codes (e.g., F3D) are shared memory codes using OPENMP or its predecessors. The remaining codes use message passing (mostly **MPI**, but one **PVM** code was tested as well). Additionally, a limited number of runs were made with the CTH code when using processors on more than one Sun HPC 10000. While most of these codes ran well, some codes did require modifications. Additionally, in the process of making these measurements, the authors gained useful insights as to what does and does not work well on these systems.

# Acknowledgments

The authors thank Dale Shires, Raju Namburu, and Ram Mohan of the U.S. Army Research Laboratory for their input and Marek Behr, formerly of the U.S. Army High Performance Computing Research Center (AHPCRC), for sharing his results. We also thank our many colleagues who worked with us over the years on our research projects, helping us to collect these data and prepare this report. We would also like to thank the employees of Business Plus, especially Claudia Coleman and Maria Brady, who assisted in the preparation and editing of this report.

Special thanks to Tom Kendall, Denice Brown, and the entire systems staff at the ARL-**MSRC** for their support of the various projects for which these runs were originally done.

This work was made possible through a grant of computer time by the Department of Defense (DOD) High Performance Computing Modernization (HPCM) Program. Additionally, some of the results mentioned in this work came from projects that were funded as part of the Common High Performance Computing Software Support Initiative (CHSSI) administered by the DOD HPCM Program.

Note: Definitions in boldface text can be found in the Glossary.

INTENTIONALLY LEFT BLANK.

# Contents

INTENTIONALLY LEFT BLANK.

# List of Figures

INTENTIONALLY LEFT BLANK.

## List of Tables

INTENTIONALLY LEFT BLANK.

# 1. Introduction

Several supercomputer architectures are viable today. **MPPs**, such as the Cray T3E, offer a large number of processors, each with its own nonshared memory. In MPP machines, when one processor needs to access data in the memory of another processor, the processor that "owns" the data must explicitly send the data to the requesting processor.*

In contrast to distributed memory architectures are shared multiprocessor **SMP** machines, such as the Sun E10000, which share memory among all the processors. In between these two examples is the SMP cluster (such as an IBM SP). Here, a small number (e.g., 2–16 in the various implementations of the IBM SPs configured with SMP nodes) of processors share memory, and the machine is made up of a large number of these SMP nodes. As in more traditional MPPs, explicit cooperation between two processors is required to transfer data from one SMP node to another.

Another intermediate architecture is the **cc-NUMA** machine, such as the SGI Origin 2000, where all the memory is logically shared but physically distributed. Here, two processors (one node) share local memory, but any processor can access all memory locations in the machine without the aid of any other processor. There can be significant differences in the designs and implementations of this class of system from vendor to vendor. As a result, some systems are much better suited for certain classes of problems—systems from SGI are heavily marketed in the scientific computing market, while systems from HP, Sequent, and Data General are more frequently marketed to the commercial/database market.

Several programming models exist today, and each is supported on one or more computer architectures. While **MPI** was developed for distributed memory machines (MPPs), it can and has been implemented on SMP and shared memory machines. Writing shared memory code is perhaps easier than writing MPI code. But many codes today are written in MPI due to the popularity of the MPP machines for the last several years. When an MPI version of a code already exists, the programmer might as well consider using it, even if it would not be their choice if writing the code from scratch. So then it becomes a performance question as to whether a shared memory version or an MPI version of the code is

* When using SHMEM (or equivalent) calls on systems that support them, programs may explicitly instruct processors to either put data into the memory of other nodes, or get data from the memory of remote nodes. However, this is very different from cache-coherent shared-memory symmetric multiprocessors, where the data resides in a globally accessible/coherent memory system accessed automatically using standard load and store instructions.

most suitable on a non-MPP machine that provides efficient support for MPI, as almost all machines now do.

As the performance runs presented in this report show, no single machine has a monopoly on the best performance with all programming models. While the Cray T3E does very well on MPI codes, it cannot run most shared memory codes. While some other machines can run all programming models, their performance varies, with each machine performing best on one code or another.

The purpose of this report is not to explain the results or conclude that one machine is better than another. Rather, its sole purpose is to document the results that different groups have reported, so readers are better equipped to come to their own conclusions about the merits of the hardware, programming paradigms, and other related issues. Furthermore, while some of the codes mentioned in this report were tuned for one or more of these machines, tuning can be a major undertaking. As a result, for **HPC** codes that are commercially available and/or maintained by other sites, the authors have little or no ability to tune them for the specific machines. Instead, the authors of those codes tuned their own codes.

The authors made these measurements as unbiasedly as possible. In fact, many of these results came from benchmarking efforts associated with procurement efforts (all such data reported in this report came from runs done in-house). Additionally, selecting which results to report was based on the perceived importance and merits of the codes in question; no results were excluded from this report because they violated a preconceived notion. As such, there are examples of different machines excelling for different codes. Some readers may wish to consider issues such as cost effectiveness, but this report does not include any cost data. Most likely, the faster machine is not always the most cost effective.

Other issues not addressed in this report or only briefly addressed are as follows:

(1) the stability of systems,

(2) the scalability of systems to very large numbers of processors,

(3) problems with the compilers and/or the operating systems,

(4) the relative merits of the input/output (I/O) systems,

(5) issues involving the queuing of jobs,

(6) the requirements of the highly varied user community that uses the resources provided by the DOD HPCM Program, and

(7) performance, profiling, and debugging tools.

## 2. Brief Observations

The following observations have been collected from a number of sources.

- **HPF** runs better on the SGI Origin than on the IBM SP (Wierschke 1997).
- HPF runs best on the Cray T3E since the Portland Group first implements new ideas on it (Shires 2000).
- In theory, jobs that run well on the SGI Origin should run well on the Sun HPC 10000. In practice, some codes would not compile, others would not run (at first), and many required some degree of tuning.
- Care should be taken to avoid "overloading" (more processes/threads actively running than there are processors) any of the shared memory systems, since overloading can result in significant performance degradation and a significant increase in **CPU** time.
- By itself, automatic parallelization is frequently of limited value; however, it may improve the performance of some programs parallelized using compiler directives.
- Many codes run well on either the Sun or SGI systems, showing reasonable levels of performance and scalability.
- Some codes will show significantly better per processor and/or overall performance on the SGI Origin than on either the Cray T3E or the IBM SP with P2SC processors.
- The performance of the Sun HPC 10000 is frequently reported to be between that of the SGI Origin 2000 with 300-MHz R12000 processors and the SGI Origin 2000 with 195-MHz R10000 processors.
- For some vectorizable codes, the shared memory programming paradigm is an excellent choice for parallelizing programs that are difficult to parallelize.
- For some codes, HPF is still the most natural programming paradigm (Mohan 1999).
- For projects requiring high levels of scalability (e.g., 128 or more processors), the IBM SP or the Cray T3E are better choices (Namburu 1999).
- Large MPPs tend to have stability problems; 128-processor Origins are particularly susceptible to periods of instability.

- Some performance differences are caused by design tradeoffs. The data show that some of these design tradeoffs sacrifice efficiency for peak speed and vice versa. Both approaches are of value and need to be considered when evaluating the merits of different systems.

## 3. Performance

Figures 1–6 and Tables 1–8 show performance results from various sources. Some of these runs were made explicitly for benchmarking the performance of a particular system, other runs were made as part of a porting/tuning effort, and a few of the runs were made for other reasons. As such, there has been no systematic attempt made to identify the reasons why a particular code runs faster on one machine than another. The authors assume that in some cases, additional tuning could improve the performance of a particular code on a particular machine. However, such tuning is beyond the scope of this report. Furthermore, when a code is not locally written/maintained, there may be little or no opportunity for the user to tune a code.

In the following CTH benchmark runs for Figure 1 and Table 2, the number of computational cells was increased by a factor of 2 each time the number of processors was doubled. This was done to maintain a constant number of computational cells per processor, which keeps the computation to communication ratio constant. In this set of benchmarks, the number of iterations was fixed, meaning that perfect scaling results in constant benchmark run times. The difference in the run time on the 64-processor Origin 2000 and the 128-processor Origin 2000 is the direct result of the increase in the average memory latency as one increases the size of an Origin 2000.

For the runs in Figure 2 and Table 3, the grid was incrementally refined by decreasing the characteristic cell length in each direction by the cubed root of two each time the number of processors doubled. In these runs, the number of iterations was not fixed. Instead, the number of iterations approximately increased by the cube root of two each time the number of processors doubled, since the time step decreases as a result of finer mesh. When ideal scaling occurs, the *Grind Time* will decrease by half every time the number of processors is doubled. This results from the units of Grind Time being microseconds/zone/cycle. Since the time per cycle is expected to remain constant and the amount of work per cycle doubled, the amount of time/zone/cycle should be halved. The amount of time/cycle should remain constant, as in Table 2. It is worthwhile noting how closely the performance of these runs matches the ideal performance. Additionally, the performance of the 300-MHz Origin 2000 and the 400-MHz Sun HPC 10000 is very similar for both these runs and those involving F3D (see Figures 3 and 4 and Tables 4 and 5).

Figures 3 and 4 and Tables 4–6 show the performance of two different versions of the implicit CFD code F3D for three problem sizes. The problem sizes range from 1-million to 206-million grid points without a significant decrease in the per processor performance. This is an indication that it is possible to tune an HPC code for a cache-based architecture. Tables 7 and 8 contain results for two other CFD codes.

Figures 5 and 6 and Tables 9 and 10 demonstrate the effect on performance and the waste of CPU time that can occur when an SMP becomes overloaded. The program used for these measurements was the shared memory version of F3D. It ran the 1-million grid point test case.

## 4. Summary

We have provided a number of observations and performance data from a variety of sources for a number of representative codes. These codes were run on the SGI Origin 2000 and the Sun HPC 10000. In many cases, there were also runs made on other commonly used HPC systems. Additionally, some of the tables provide comparisons of the performance achievable when using various programming paradigms. The last two tables demonstrate the inefficiency of allowing an SMP to become overloaded. It is hoped that this report and, in particular, the figures and data tables will enable the reader to better evaluate the merits of these systems in relation to his or her needs.

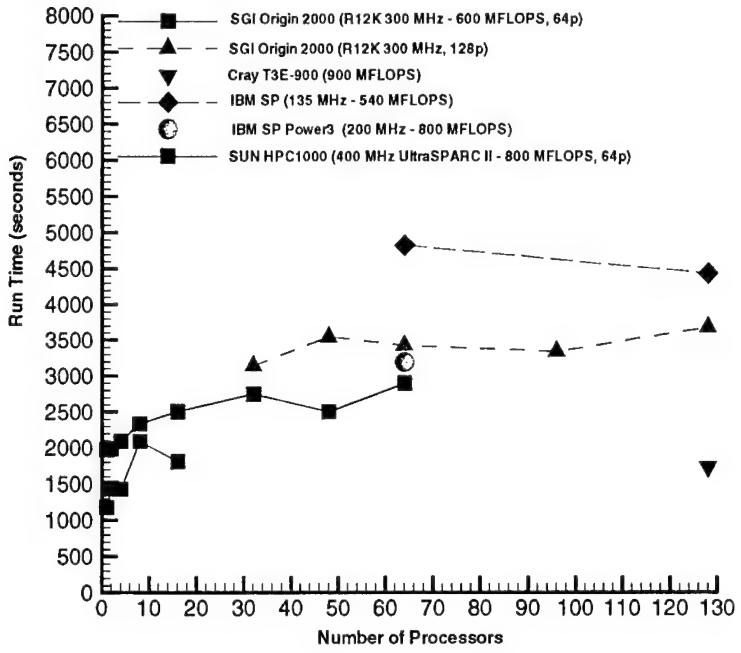



Figure 1. CTH run times scaling the data set size in proportion to the number of processors used (data set supplied by Raju Namburu of the U.S. Army Research Laboratory, Aberdeen Proving Ground, MD).

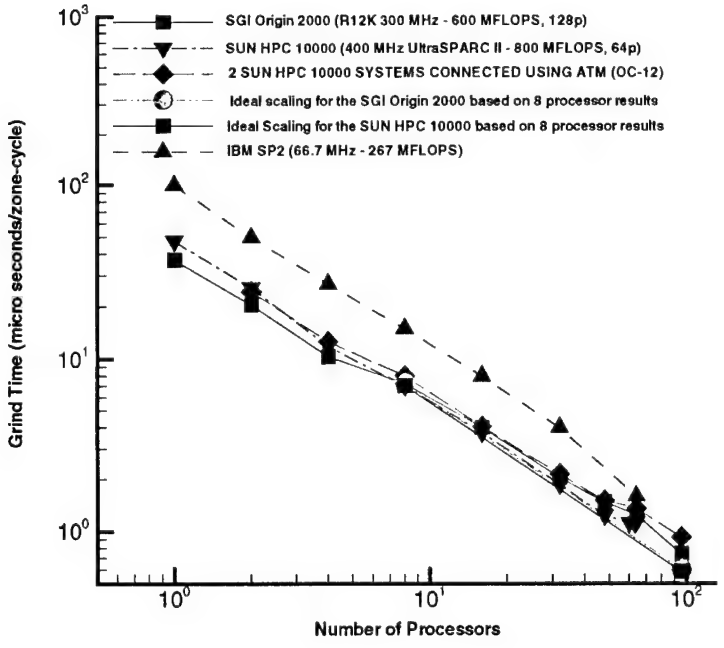



Figure 2. CTH run times (data set supplied by Steve Schraml of the U.S. Army Research Laboratory, Aberdeen Proving Ground, MD).

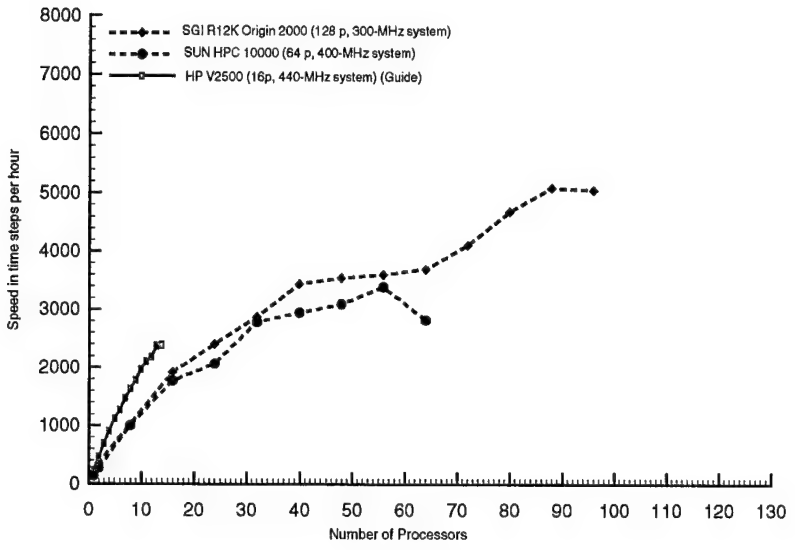



Figure 3a. The performance of the shared memory version of the F3D code when run on modern scalable SMPs (1-million grid point test case).*

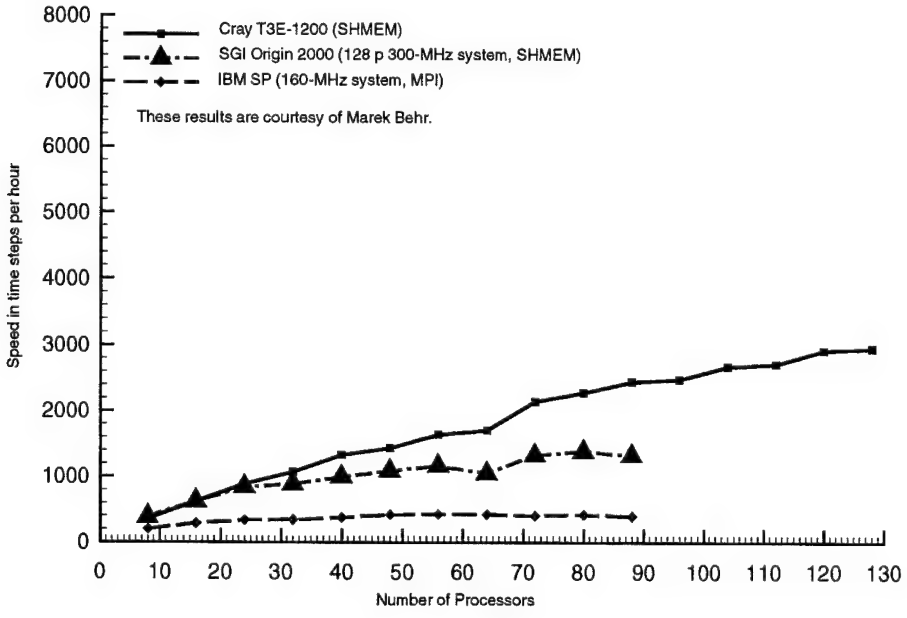



Figure 3b. The performance of the distributed memory version of the F3D code when run on a modern scalable SMP/MPPs (1-million grid point test case).*

* The speeds have been adjusted to remove startup and termination costs.

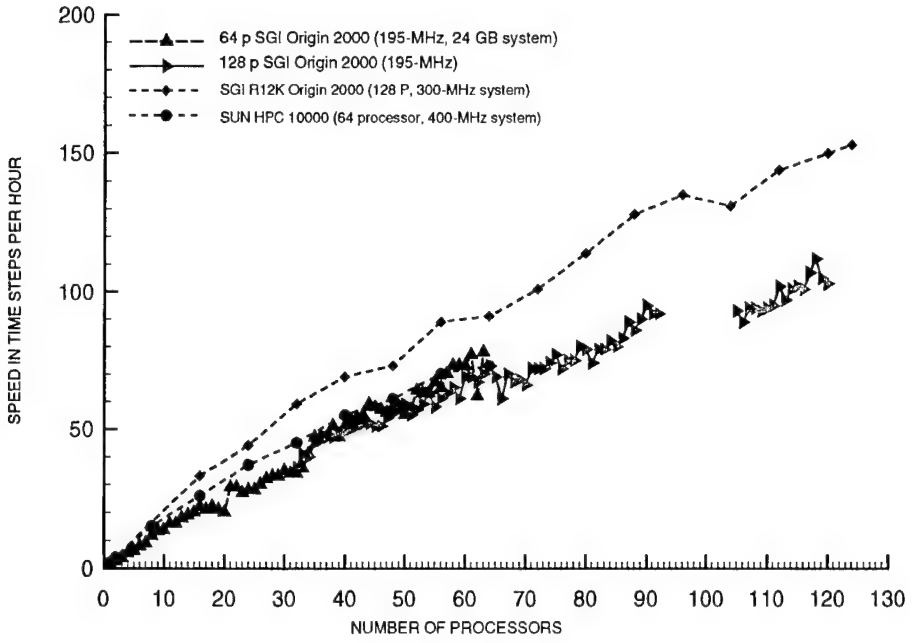



Figure 4. The performance of the shared memory version of the F3D code when run on modern scalable SMPs (59-million grid point test case).*

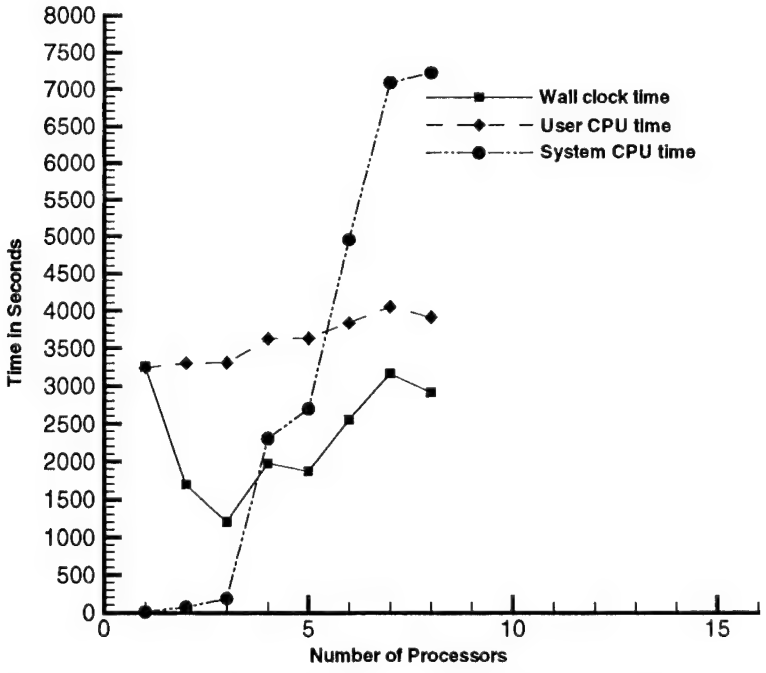



Figure 5. The effect on performance and the consumption of CPU time from running a parallel job on an overloaded HP V-Class.

* The speeds have been adjusted to remove startup and termination costs.

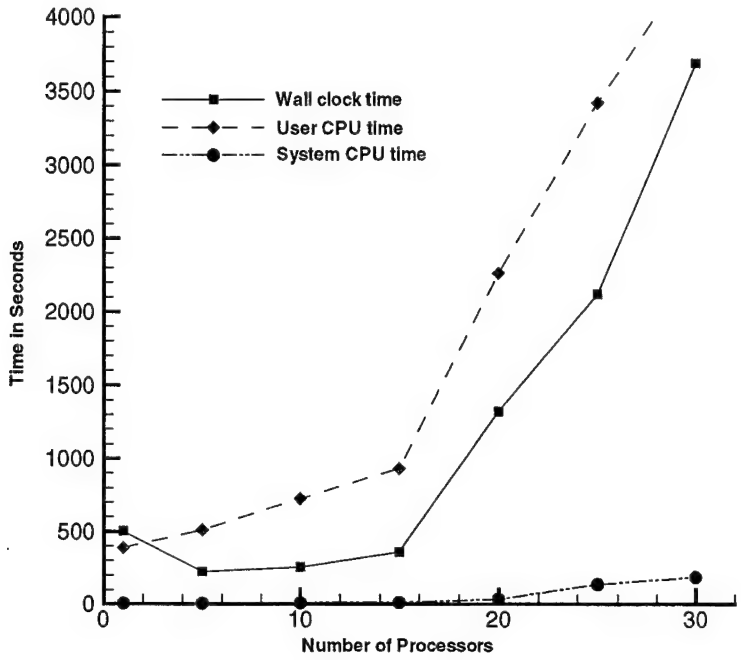



Figure 6. The effect on performance and the consumption of CPU time from running a parallel job on an overloaded SGI Origin 2000.

Table 1. Miscellaneous benchmarking runs.

| Program/Dataset | SGI (300-MHz R12000 Origin) (hh:mm/no. of processors) | Sun (400-MHz UltraSPARC II) (hh:mm/no. of processors) |
|---|---|---|
| Gaussian 98 | Ran | Failed to run |
| Overflow (MPI version) | 2:40/24 | 3:25/24 |
| CTH/128.in | 6:58/64<br>7:31/56 | 6:14/64 |
| POP | 3:18/16 | Failed to compile |
| Gamess | 0:19/12 | 0:12/12 |
| Xpatch | 4:23/1 | 6:23/1 |

Table 2. CTH benchmark runs.[a, b]

| System | Processor Speed (MHz) | Peak Performance (MFLOPS) | No. of Processors | Run Time (s) |
|---|---|---|---|---|
| SGI Origin 2000 (64-processor system) | 300 | 600 | 1 | 1178 |
| | | | 2 | 1439 |
| | | | 4 | 1427 |
| | | | 8 | 2089 |
| | | | 16 | 1811 |
| SGI Origin 2000 (128-processor system) | 300 | 600 | 32 | 3144 |
| | | | 48 | 3544 |
| | | | 64 | 3423 |
| | | | 96 | 3339 |
| | | | 128 | 3676 |
| Cray T3E-900 | 450 | 900 | 128 | 1732 |
| IBM SP (P2SC) | 135 | 540 | 64 | 4822 |
| | | | 128 | 4433 |
| Sun HPC 10000 | 400 | 800 | 1 | 1971 |
| | | | 2 | 1986 |
| | | | 4 | 2092 |
| | | | 8 | 2331 |
| | | | 16 | 2506 |
| | | | 32 | 2749 |
| | | | 48 | 2501 |
| | | | 64 | 2895 |
| Sun HPC 10000 (96-processor dataset) | 400 | 800 | 64 | 4391 |
| Sun HPC 10000 (128-processor dataset) | 400 | 800 | 64 | 5673 |

[a] Data set courtesy of Raju Namburu, U.S. Army Research Laboratory, Aberdeen Proving Ground, MD.
[b] The job size was scaled in proportion to the number of processors.

Table 3. Additional CTH results.[a, b]

| System | Processor Speed (MHz) | Peak Performance (MFLOPS) | No. of Processors | Grind Time (μs/zone/cycle) | |
|---|---|---|---|---|---|
| | | | | Actual | Ideal[b] |
| SGI Origin 2000 (128 processor) | 300 | 600 | 1 | 36.979 | NA |
| | | | 2 | 20.479 | NA |
| | | | 4 | 10.355 | NA |
| | | | 8 | 7.2749 | 7.2749 |
| | | | 16 | 4.0035 | 3.6375 |
| | | | 32 | 2.0599 | 1.8187 |
| | | | 48 | 1.4815 | 1.2125 |
| | | | 64 | 1.2456 | 0.90936 |
| | | | 96 | 0.73997 | 0.60624 |
| SGI Origin 2000 (128 processor) | 195 | 390 | 1 | 53.155 | NA |
| Sun HPC 10000 | 400 | 800 | 1 | 47.558 | NA |
| | | | 2 | 25.622 | NA |
| | | | 4 | 11.875 | NA |
| | | | 8 | 7.0330 | 7.0330 |
| | | | 16 | 3.7468 | 3.5165 |
| | | | 32 | 1.8792 | 1.7583 |
| | | | 48 | 1.2385 | 1.1722 |
| | | | 60 | 1.1170 | 0.93773 |
| | | | 63 | 1.1075 | 0.89308 |
| | | | 64 | 1.1332 | 0.87913 |
| 2 Sun HPC 1000 connected using ATM (OC-12) | 400 | 800 | 2 | 24.357 | NA |
| | | | 4 | 12.635 | NA |
| | | | 8 | 8.0182 | 8.0182 |
| | | | 16 | 4.0605 | 4.0091 |
| | | | 32 | 2.1539 | 2.0046 |
| | | | 48 | 1.5136 | 1.3364 |
| | | | 64 | 1.3593 | 1.0023 |
| | | | 96 | 0.92424 | 0.66818 |
| IBM SP (Power 2) | 66.7 | 267 | 1 | 100.24 | NA |
| | | | 2 | 50.12 | NA |
| | | | 4 | 26.83 | NA |
| | | | 8 | 15.23 | 15.230 |
| | | | 16 | 8.13 | 7.615 |
| | | | 32 | 4.09 | 3.808 |
| | | | 64 | 1.69 | 1.904 |

[a] The job size was scaled in proportion to the number of processors (Kimsey et al. 1998; Schraml and Kimsey 2000).

[b] The ideal values are extrapolated from the performance of runs using eight processors.

Table 4. The performance of various versions of the F3D code when run on modern scalable systems (1-million grid point test case).[a]

| System | Peak Processor Speed (MFLOPS) | No. of Processors Used | Version | Speed | |
|---|---|---|---|---|---|
| | | | | (time steps/hr) | MFLOPS |
| SGI R10K O2K | 390 | 8 | Compiler Directives | 793 | 1.04E3 |
| SGI R12K O2K | 600 | 8 | SHMEM | 382 | 4.99E2 |
| SGI R10K O2K | 390 | 32 | Compiler Directives | 2138 | 2.79E3 |
| SGI R12K O2K | 600 | 32 | SHMEM | 989 | 1.29E3 |
| | 600 | | Compiler Directives | 2877 | 3.76E3 |
| SGI R10K O2K | 390 | 48 | Compiler Directives | 2725 | 3.56E3 |
| SGI R12K O2K | 600 | 48 | SHMEM | 1083 | 1.42E3 |
| | 600 | | Compiler Directives | 3545 | 4.63E3 |
| SGI R10K O2K | 390 | 64 | Compiler Directives | 2601 | 3.40E3 |
| SGI R12K O2K | 600 | 64 | SHMEM | 1050 | 1.37E3 |
| | 600 | | Compiler Directives | 3694 | 4.83E3 |
| SGI R10K O2K | 390 | 88 | Compiler Directives | 3619 | 4.73E3 |
| SGI R12K O2K | 600 | 88 | SHMEM | 1320 | 1.73E3 |
| | 600 | | Compiler Directives | 5087 | 6.65E3 |
| Cray T3E-1200 | 1200 | 8 | SHMEM | 349 | 4.56E2 |
| | | 32 | | 1062 | 1.39E3 |
| | | 48 | | 1431 | 1.87E3 |
| | | 64 | | 1705 | 2.23E3 |
| | | 88 | | 2443 | 3.19E3 |
| | | 128 | | 2948 | 3.85E3 |
| IBM SP 160 (MHz) | 640 | 8 | MPI | 199 | 2.60E2 |
| | | 32 | | 342 | 4.47E2 |
| | | 48 | | 420 | 5.49E2 |
| | | 64 | | 423 | 5.52E2 |
| | | 88 | | 396 | 5.18E2 |
| Sun HPC 10000 | 800 | 8 | Compiler Directives | 999 | 1.31E3 |
| | | 32 | | 2619 | 3.64E3 |
| | | 48 | | 3093 | 4.04E3 |
| | | 56 | | 3391 | 4.43E3 |
| | | 64 | | 2819 | 3.68E3 |
| HP V-Class | 1760 | 8 | Compiler Directives | 1632 | 2.13E3 |
| | | 14 | | 2392 | 3.13E3 |

[a] For additional details, see Behr et al. (2000).

Table 5. The performance of the shared memory version of the F3D code when run on modern scalable SMPs (59-million grid point test case).

| System | Peak Processor Speed (MFLOPS) | No. of Processors Used | Speed | |
|---|---|---|---|---|
| | | | (time steps/hr) | MFLOPS |
| SGI R12K Origin | 600 | 1 | 2.3 | 1.79E2 |
| | | 16 | 33 | 2.57E3 |
| | | 32 | 59 | 4.59E3 |
| | | 48 | 73 | 5.68E3 |
| | | 64 | 91 | 7.08E3 |
| | | 96 | 135 | 1.05E4 |
| | | 124 | 153 | 1.19E4 |
| Sun HPC 10000 | 800 | 1 | 2.1 | 1.63E2 |
| | | 8 | 15.1 | 1.18E3 |
| | | 16 | 26 | 2.02E3 |
| | | 32 | 45 | 3.50E3 |
| | | 48 | 61 | 4.75E3 |
| | | 56 | 70 | 5.45E3 |
| | | 64 | 73 | 5.68E3 |

Table 6. The performance of the shared memory version of the F3D code when run on modern scalable SMPs (206-million grid point test case).

| System | Peak Processor Speed (MFLOPS) | No. of Processors Used | Speed | |
|---|---|---|---|---|
| | | | (time steps/hr) | MFLOPS |
| SGI R12K Origin | 600 | 1 | 0.62 | 1.67E2 |
| | | 16 | 7.4 | 2.00E3 |
| | | 32 | 15.2 | 4.10E3 |
| | | 48 | 18 | 4.86E3 |
| | | 64 | 26 | 2.02E3 |
| | | 96 | 38 | 1.03E4 |
| | | 124 | 48 | 1.30E4 |

Table 7. A comparison of the performance of the shared memory implementation of the CFD code Overflow and the PVM implementation of the same code.[a]

| System | Peak Processor Speed (MFLOPS) | No. of Processors Used | Run Time (s) | |
|---|---|---|---|---|
| | | | Shared Memory[b] | PVM |
| SGI R10K Origin | 390 | 1 | 959 | N/A |
| | | 4 | 318 | 335 |
| | | 8 | 184 | 191 |
| | | 16 | 129 | 117 |
| | | 31 | 96 | N/A[c] |

[a] For a complete discussion of these results, see Hisley et al. (1998).

[b] The shared memory implementation combined compiler directives and the automatic parallelization facility (-pfa).

[c] The 31-processor PVM run was not made because it was too difficult to decompose the grids with the available tools.

Table 8. The performance of LES (a CFD code using direct simulation of large eddies).[a, b]

| System | Peak Processor Speed (MFLOPS) | No. of Processors Used | Run Time (s) |
|---|---|---|---|
| SGI R12K Origin | 600 | 1 | 1232 |
| | | 2 | 619 |
| | | 4 | 314 |
| | | 16 | 153 |

[a] 64 × 64 grid.
[b] The program was parallelized using the SPMD programming style with OpenMP.

Table 9. The effect on performance and the consumption of CPU time from running a parallel job on an overloaded HP V-Class.[a]

| No. of Processors Used | Wall Clock Time (s) | User CPU Time (s) | System CPU Time (s) |
|---|---|---|---|
| 1 | 3524 | 3244 | 8 |
| 2 | 1698 | 3301 | 72 |
| 3 | 1203 | 3303 | 186 |
| 4 | 1974 | 3625 | 2302 |
| 5 | 1871 | 3630 | 2696 |
| 6 | 2554 | 3837 | 4955 |
| 7 | 3166 | 4051 | 7089 |
| 8 | 2915 | 3915 | 7223 |

[a] The job was run for 200 time steps.

Table 10. The effect on performance and the consumption of CPU time from running a parallel job on an overloaded SGI Origin 2000.[a]

| No. of Processors Used | Wall Clock Time (s) | User CPU Time (s) | System CPU Time (s) |
|---|---|---|---|
| 1 | 503 | 390 | 5 |
| 5 | 225 | 512 | 7 |
| 10 | 256 | 729 | 9 |
| 15 | 360 | 935 | 11 |
| 20 | 1322 | 2263 | 36 |
| 25 | 2119 | 3423 | 138 |
| 30 | 3691 | 4414 | 188 |

[a] The job was run for 40 time steps.

# 5. References


Behr, M., D. M. Pressel, and W. B. Sturek, Sr. "Comments on CFD Code Performance on Scalable Architectures." *Computer Methods in Applied Mechanics*, New York: Elsevier Science LTD, 2000.

Hisley, D. M., G. Agrawal, and L. Pollock. "Performance Studies of the Parallelization of a CFD Solver on the Origin 2000." Proceedings of the 21st Army Science Conference, Department of the Army, 1998.

Kimsey, K. D., S. J. Schraml, and E. S. Hertel. "Scalable Computation in Penetration Mechanics." *International Journal on Advances in Engineering Software Including Computing Systems in Engineering*, vol. 29, pp. 209–215, 1998.

Mohan, R. Personal communication with D. Pressel. U.S. Army Research Laboratory, Aberdeen Proving Ground, MD, 1999.

Namburu, R. Personal communication with D. Pressel. U.S. Army Research Laboratory, Aberdeen Proving Ground, MD, 1999.

Schraml, S. J., and K. D. Kimsey. "Scalability of the CTH Hydrodynamics Code on the HPC 10000 Architecture." ARL-TR-2173, U.S. Army Research Laboratory, Aberdeen Proving Ground, MD, February 2000.

Shires, D. Personal communication with D. Pressel. U.S. Army Research Laboratory, Aberdeen Proving Ground, MD, 2000.

Wierschke, S. G., MAJ. "CHSSI Semiannual Report: Computational Chemistry and Materials Science (CCM)." U.S. Air Force Research Laboratory, 15 October 1997.

INTENTIONALLY LEFT BLANK.

## Glossary

| | |
|---|---|
| cc-NUMA | Cache coherent nonuniform memory access |
| CPU | Central Processing Unit |
| CTA | Computational Technology Area |
| DC | Distributed Center |
| HPC | High-Performance Computing |
| HPF | High Performance Fortram |
| MFLOPS | Million Floating Point Operations Per Second |
| MPI | Message Passing Interface |
| MPP | Massively Parallel Processor |
| MSRC | Major Shared Resource Center |
| PVM | Parallel Virtual Machine |
| SHMEM | Low latency message passing library developed by CRAY Research for the T3D and T3E product lines. |
| SMP | Symmetric Multiprocessor—a term normally only applied to shared memory systems using hardware memory coherency protocols. |
| SPMD | Single Program Multiple Data |

INTENTIONALLY LEFT BLANK.

| NO. OF COPIES | ORGANIZATION |
|---|---|
| 2 | DEFENSE TECHNICAL<br>INFORMATION CENTER<br>DTIC OCA<br>8725 JOHN J KINGMAN RD<br>STE 0944<br>FT BELVOIR VA 22060-6218 |
| 1 | HQDA<br>DAMO FDT<br>400 ARMY PENTAGON<br>WASHINGTON DC 20310-0460 |
| 1 | OSD<br>OUSD(A&T)/ODDR&E(R)<br>DR R J TREW<br>3800 DEFENSE PENTAGON<br>WASHINGTON DC 20301-3800 |
| 1 | COMMANDING GENERAL<br>US ARMY MATERIEL CMD<br>AMCRDA TF<br>5001 EISENHOWER AVE<br>ALEXANDRIA VA 22333-0001 |
| 1 | INST FOR ADVNCD TCHNLGY<br>THE UNIV OF TEXAS AT AUSTIN<br>3925 W BRAKER LN STE 400<br>AUSTIN TX 78759-5316 |
| 1 | US MILITARY ACADEMY<br>MATH SCI CTR EXCELLENCE<br>MADN MATH<br>THAYER HALL<br>WEST POINT NY 10996-1786 |
| 1 | DIRECTOR<br>US ARMY RESEARCH LAB<br>AMSRL D<br>DR D SMITH<br>2800 POWDER MILL RD<br>ADELPHI MD 20783-1197 |
| 1 | DIRECTOR<br>US ARMY RESEARCH LAB<br>AMSRL CI AI R<br>2800 POWDER MILL RD<br>ADELPHI MD 20783-1197 |

| NO. OF COPIES | ORGANIZATION |
|---|---|
| 3 | DIRECTOR<br>US ARMY RESEARCH LAB<br>AMSRL CI LL<br>2800 POWDER MILL RD<br>ADELPHI MD 20783-1197 |
| 3 | DIRECTOR<br>US ARMY RESEARCH LAB<br>AMSRL CI IS T<br>2800 POWDER MILL RD<br>ADELPHI MD 20783-1197 |
| | ABERDEEN PROVING GROUND |
| 2 | DIR USARL<br>AMSRL CI LP (BLDG 305) |

| NO. OF COPIES | ORGANIZATION |
|---|---|
| 1 | PROGRAM DIRECTOR<br>C HENRY<br>1010 N GLEBE RD STE 510<br>ARLINGTON VA 22201 |
| 1 | DPTY PROGRAM DIRECTOR<br>L DAVIS<br>1010 N GLEBE RD STE 510<br>ARLINGTON VA 22201 |
| 1 | DISTRIBUTED CENTERS<br>PROJECT OFFICER<br>V THOMAS<br>1010 N GLEBE RD STE 510<br>ARLINGTON VA 22201 |
| 1 | HPC CTRS PROJECT MNGR<br>J BAIRD<br>1010 N GLEBE RD STE 510<br>ARLINGTON VA 22201 |
| 1 | CHSSI PROJECT MNGR<br>L PERKINS<br>1010 N GLEBE RD STE 510<br>ARLINGTON VA 22201 |
| 1 | RICE UNIVERSITY<br>MECHANICAL ENGRNG &<br>MATERIALS SCIENCE<br>M BEHR MS 321<br>6100 MAIN ST<br>HOUSTON TX 77005 |
| 1 | J OSBURN CODE 5594<br>4555 OVERLOOK RD<br>BLDG A49 RM 15<br>WASHINGTON DC 20375-5340 |
| 1 | NAVAL RSCH LAB<br>J BORIS CODE 6400<br>4555 OVERLOOK AVE SW<br>WASHINGTON DC 20375-5344 |
| 1 | WL FIMC<br>B STRANG<br>BLDG 450<br>2645 FIFTH ST STE 7<br>WPAFB OH 45433-7913 |
| 1 | NAVAL RSCH LAB<br>R RAMAMURTI CODE 6410<br>WASHINGTON DC 20375-5344 |

| NO. OF COPIES | ORGANIZATION |
|---|---|
| 1 | ARMY AEROFLIGHT<br>DYNAMICS DIRECTORATE<br>R MEAKIN M S 258 1<br>MOFFETT FIELD CA 94035-1000 |
| 1 | NAVAL RSCH LAB<br>HEAD OCEAN DYNAMICS<br>& PREDICTION BRANCH<br>J W MCCAFFREY JR CODE 7320<br>STENNIS SPACE CENTER MS<br>39529 |
| 1 | US AIR FORCE WRIGHT LAB<br>WL FIM<br>J J S SHANG<br>2645 FIFTH ST STE 6<br>WPAFB OH 45433-7912 |
| 1 | US AIR FORCE PHILIPS LAB<br>OLAC PL RKFE<br>CAPT S G WIERSCHKE<br>10 E SATURN BLVD<br>EDWARDS AFB CA 93524-7680 |
| 1 | NAVAL RSCH LAB<br>DR D PAPACONSTANTOPOULOS<br>CODE 6390<br>WASHINGTON DC 20375-5000 |
| 1 | AIR FORCE RSCH LAB DEHE<br>R PETERKIN<br>3550 ABERDEEN AVE SE<br>KIRTLAND AFB NM 87117-5776 |
| 1 | NAVAL RSCH LAB<br>RSCH OCEANOGRAPHER CNMOC<br>G HEBURN<br>BLDG 1020 RM 178<br>STENNIS SPACE CENTER MS<br>39529 |
| 1 | AIR FORCE RSCH LAB<br>INFORMATION DIRECTORATE<br>R W LINDERMAN<br>26 ELECTRONIC PKWY<br>ROME NY 13441-4514 |
| 1 | SPAWARSYSCEN D4402<br>R A WASILAUSKY<br>BLDG 33 RM 0071A<br>53560 HULL ST<br>SAN DIEGO CA 92152-5001 |

NO. OF COPIES ORGANIZATION

1 USAE WATERWAYS
EXPERIMENT STATION
CEWES HV C
J P HOLLAND
3909 HALLS FERRY RD
VICKSBURG MS 39180-6199

1 US ARMY CECOM RSCH
DEVELOPMENT & ENGRNG CTR
AMSEL RD C2
B S PERLMAN
FT MONMOUTH NJ 07703

1 SPACE & NAVAL WARFARE
SYSTEMS CTR
K BROMLEY CODE D7305
BLDG 606 RM 325
53140 SYSTEMS ST
SAN DIEGO CA 92152-5001

1 DIRECTOR
DEPARTMENT OF ASTRONOMY
P WOODWARD
356 PHYSICS BLDG
116 CHURCH ST SE
MINNEAPOLIS MN 55455

1 RICE UNIVERSITY
MECHANICAL ENGRNG &
MATERIALS SCIENCE
T TEZDUYAR MS 321
6100 MAIN ST
HOUSTON TX 77005

1 ARMY HIGH PERFORMANCE
COMPUTING RSCH CTR
B BRYAN
1200 WASHINGTON AVE
S MINNEAPOLIS MN 55415

1 ARMY HIGH PERFORMANCE
COMPUTING RSCH CTR
G V CANDLER
1200 WASHINGTON AVE
S MINNEAPOLIS MN 55415

1 NAVAL CMD CNTRL &
OCEAN SURVEILLANCE CTR
L PARNELL
NCCOSC RDTE DIV D3603
49590 LASSING RD
SAN DIEGO CA 92152-6148

NO. OF COPIES ORGANIZATION

1 UNIVERSITY OF TENNESSEE
COMPUTER SCIENCE DEPT
S MOORE
1122 VOLUNTEER BLVD
STE 203
KNOXVILLE TN 37996-3450

ABERDEEN PROVING GROUND

33 DIR USARL
AMSRL CI
N RADHAKRISHNAN
AMSRL CI H
C NIETUBICZ
W STUREK
AMSRL CI HC
P CHUNG
J CLARKE
D HISLEY
M HURLEY
A MARK
R MOHAN
R NAMBURU
D PRESSEL
D SHIRES
R VALISETTY
C ZOLTANI
AMSRL CI HS
D BROWN
T KENDALL
M KNOWLES
P MATTHEWS
R PRABHAKARAN
T PRESSLEY
K SMITH
S THOMPSON
AMSRL WM BC
K HEAVEY
J SAHU
P WEINACHT
AMSRL WM BF
H EDGE
AMSRL WM T
B BURNS
AMSRL WM TA
D KLEPONIS
M NORMANDIA
AMSRL WM TC
R COATES
K KIMSEY
S SCHETTLER
S SCHRAML

INTENTIONALLY LEFT BLANK.

# REPORT DOCUMENTATION PAGE

*Form Approved*
*OMB No. 0704-0188*



| 1. AGENCY USE ONLY *(Leave blank)* | 2. REPORT DATE | 3. REPORT TYPE AND DATES COVERED |
|---|---|---|
| | March 2002 | Final, July 1999–June 2000 |

**4. TITLE AND SUBTITLE**
A Comparison of the Performance of Two Popular Symmetric Multiprocessors When Used to Run High Performance Computing Applications

**5. FUNDING NUMBERS**
665803.731

**6. AUTHOR(S)**
Daniel M. Pressel, Stephen Schraml, Steven Thompson,* Dixie Hisley, Punyam Satya-narayana,* Michael Knowles,* and Darren M. Wah†

**7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**
U.S. Army Research Laboratory
ATTN: AMSRL-CI-HC
Aberdeen Proving Ground, MD 21005-5067

**8. PERFORMING ORGANIZATION REPORT NUMBER**
ARL-TR-2476

**9. SPONSORING/MONITORING AGENCY NAMES(S) AND ADDRESS(ES)**

**10.SPONSORING/MONITORING AGENCY REPORT NUMBER**

**11. SUPPLEMENTARY NOTES**
* U.S. Army Research Laboratory Major Shared Resource Center, Raytheon Systems Company, 939-I Beards Hill Rd., Suite 191, Aberdeen, MD 21001 † U.S. Army Research Laboratory Major Shared Resource Center, HPTi, 939-I Beards Hill Rd., Suite 191, Aberdeen, MD 21001

**12a. DISTRIBUTION/AVAILABILITY STATEMENT**
Approved for public release; distribution is unlimited.

**12b. DISTRIBUTION CODE**

**13. ABSTRACT** *(Maximum 200 words)*
Traditionally, symmetric multiprocessors have used modest numbers of processors. Since many of them were bus-based systems, they inherently lacked scalability to what might be referred to as moderate-sized systems. With the advent of the Sun HPC 10000 and the SGI Origin, we now have symmetric multiprocessors that have successfully scaled to moderate-sized systems. In fact, SGI has had some success at scaling the Origin into the lower end of the range of large systems. The first symmetric multiprocessor to make that claim was the Convex Exemplar. But based on our experience at the Distributed Center located at NRAD, San Diego, CA (now the Naval Command Control and Ocean Surveillance Center), its overall performance and scalability left something to be desired.

This report presents the results from runs involving a variety of programs on the SGI Origins and Sun HPC 10000s located at the U.S. Army Research Laboratory (ARL)-MSRC, the Naval Research Laboratory (NRL-DC), Washington, DC, and other places. Some of these codes (e.g., F3D) are shared memory codes using OPENMP or its predecessors. The remaining codes use message passing (mostly MPI, but one PVM code was tested as well). Additionally, a limited number of runs were made with the CTH code when using processors on more than one Sun HPC 10000. While most of these codes ran well, some codes did require modifications. Additionally, in the process of making these measurements, the authors gained useful insights as to what does and does not work well on these systems.

**14. SUBJECT TERMS**
supercomputer, high performance computing, parallel programming, shared memory programming

**15. NUMBER OF PAGES**
26

**16. PRICE CODE**

| 17. SECURITY CLASSIFICATION OF REPORT | 18. SECURITY CLASSIFICATION OF THIS PAGE | 19. SECURITY CLASSIFICATION OF ABSTRACT | 20. LIMITATION OF ABSTRACT |
|---|---|---|---|
| UNCLASSIFIED | UNCLASSIFIED | UNCLASSIFIED | UL |

INTENTIONALLY LEFT BLANK.